

Lab 9

If Statements

GOALS:

1. Students will be able to create a program that involves simple if statements.
2. Students will be able to write code for an extended if statement.

ESSENTIAL QUESTIONS

1. Why are if statements necessary for a program?
2. What makes humans different than computers when it comes to decision making?

Relational Operators

Relational operators are used to form Boolean expressions that evaluates to true or false. Here is the list of relational operators: (=) → equals to, (<) → less than, (<=) → less than or equal to, (>) → greater than, (>=) greater than or equal to, (<>) → not equal to. Very similar to Algebra.

If Statements

If statements allow the computer to make decisions according to the condition set by the programmer.

Practice Create the following program

Step 1: Create the form on the right.

Step 2: Name and use the appropriate captions

Step 3: Type the code in the appropriate places

Step 4: Save & Run

Name the objects as the following:

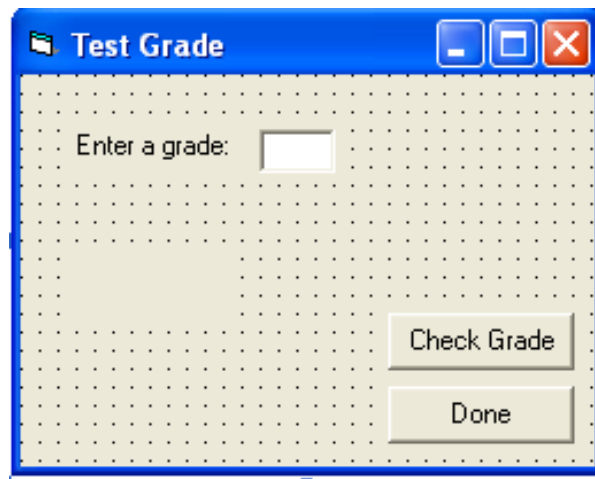
Label1 → lblDisplay

Label2 → lblMessage

TextBox → txtGrade

Command Button 1 → cmdCheckGrade

Command Button 2 → cmdDone



Code the following:

Option Explicit

```
Private Sub txtGrade_Change()
```

```
    lblMessage.Caption = ""
```

```
End Sub
```

```

Private Sub cmdCheckGrade_Click()
    Dim dblGrade As Double
    dblGrade = txtGrade.Text

    If dblGrade >= 70 Then
        lblMessage.Caption = "Good job!"
    End If
End Sub

Private Sub cmdDone_Click()
    Unload Me
End Sub

```

QUESTIONS:

1. What happens when a grade higher than 70 is entered?
2. What happens when a grade of 70 is entered?
3. What happens when a grade lower than 70 is inputted?
4. What happens if you remove the equal symbol from the statement “If dblGrade >= 70 Then”? Remove the equal symbol and run the program and enter 70, what happens?

If ...Then...Else Statements

Using the program above, add the else statement to the if statement in the above program

Option Explicit

```

Private Sub txtGrade_Change()
    lblMessage.Caption = ""
End Sub

Private Sub cmdCheckGrade_Click()
    Dim dblGrade As Double
    dblGrade = txtGrade.Text

    If dblGrade >= 70 Then
        lblMessage.Caption = "Good job!"
    Else
        lblMessage.Caption = "Study more."
    End If

```

‘ADDED CODE

End Sub

Private Sub cmdDone_Click()

Unload Me

End Sub

Step 1: Run Program

Step 2: Save

Questions

1. What happens when a grade higher than 70 is entered?
2. What happens when a grade of 70 is entered?
3. What happens when a grade lower than 70 is inputted?
4. Was there a time when both code statements ran at the same time during a run?
5. What happens when the user types a number in the text box before clicking the button?

Style – Notice the statement inside the “if – then” structure is indented. An “if statement” can have more than one line of code between the “if and end if”, but the statements should be indented.

Nested if statements

Step 1: Create the form to the right.

Step 2: Name and use the appropriate captions

Step 3: Type the code in the appropriate places

Step 4: Save & Run

Name the following objects

Label1 → lblMessage

Label2 → lblPrompt

Label3 → lblDisplay

Command1 → cmdCheckGuess

Command2 → cmdQuit

Private Sub cmdCheckGuess_Click()

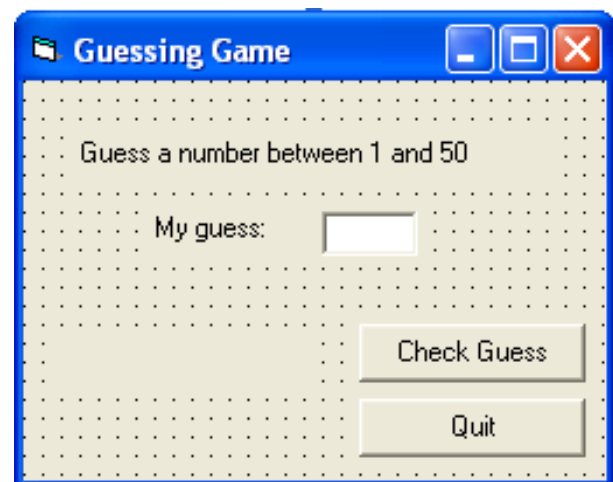
Const intSECRETNUM as Integer = 37

Dim intGuess as Integer

intGuess = txtInput.text

If intGuess = intSECRETNUM Then

lblDisplay.Caption = “You guessed it!”



```

Else
    If (intGuess < intSECRETNUM Then
        lblDisplay.Caption = "Too Low!"
    else
        lblDisplay.Caption = "Too High!"
    end if
end if
End Sub

```

QUESTIONS:

1. What condition(s) must take place for "Too High!" to run?
2. What condition(s) must take place for "Too Low!" to occur?
3. Why is it important of the indentation of the If-else-end if in the Else portion of the first if statement?

EXTENDED IF STATEMENTS

Step1: Create the following programs using the appropriate names and captions from the code below.

Option Explicit

```

Private Sub txtGuess_Change()
    lblGuessCheckedMessage.Caption = ""
End Sub

```

```

Private Sub cmdCheckGuess_Click()
    Const intSecretNum As Integer = 25
    Dim intGuess As Integer

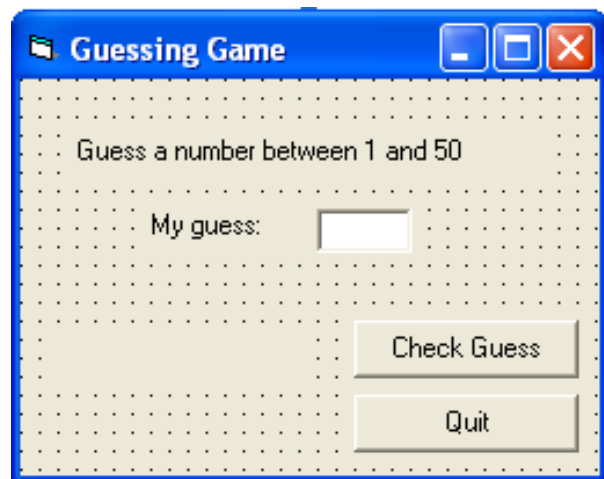
    intGuess = txtGuess.Text
    If intGuess = intSecretNum Then
        lblGuessCheckedMessage.Caption = "You guessed it!"
    ElseIf intGuess < intSecretNum Then
        lblGuessCheckedMessage.Caption = "Too low."
    Else
        lblGuessCheckedMessage.Caption = "Too high."
    End If
End Sub

```

```

Private Sub cmdQuit_Click()
    Unload Me
End Sub

```



QUESTIONS

1. What is the difference of a nested if statement and an extended if statement?
2. Can a nested if statement be rewritten as an extended if statement?
3. Why should you use extended if statements over many if statements?
4. Can more than one extended if statement be true and run at the same time?
5. How many extended if statements can be true?

Extended if statements are much more efficient than use a bunch of regular if statements. With extended if statements, the first if statement that evaluates to be true; the code runs, and **the rest of the else if statements are skipped even if they are true.** The next statement that is executed is after the end if of the extended if statement. Using a bunch of regular if statements, each if statement must be checked even if the first if statement is true.